03-0781
1496.00317

METHOD AND/OR CIRCUIT FOR BINARY ARITHMETIC DECODING DECISIONS BEFORE TERMINATION

Field of the Invention

The present invention relates to a digital video processing generally and, more particularly, to a method for binary arithmetic decoding decisions before termination.

Background of the Invention

The proposed H.264 video coding includes an I_PCM (intra-frame pulse code modulation) macroblock coding mode. While video compression techniques attempt to compress every macroblock of a video sequence, there is no guarantee that individual macroblocks will in fact be compressed. In practice, white noise will be expanded to some extent by compression techniques.

The I_PCM mode in the current H.264 specification provides a coding mode that guarantees a limit on the expansion of white noise during compression. The I_PCM mode provides a mechanism for an encoder to potentially more easily produce a bistream that guarantees a maximum number of bits per macroblock,

thereby potentially enabling simpler decoding hardware that takes advantage of the guaranteed limitation.

The I_PCM mode generates the actual values of the pixels contained in a 16 x 16 macroblock, rather than attempting to compress the information. The I_PCM mode is a type of "fail safe" mode that bounds the size of a macroblock in the compressed video bitstream. While PCM coding has been implemented in the past, H.264 is the first instance where an I_PCM macroblock mode is incorporated into a video encoder/decoder (CODEC) that switches between PCM and non-PCM coding modes.

The current H.264 proposal does not provide a provision to bypass the context adaptive arithmetic entropy encoding (CABAC) stage for I_PCM mode encoded macroblocks. An idea behind the I_PCM mode is to avoid all compression. Therefore, it would be beneficial to bypass the entropy encoding stage for bits that belong to an I_PCM mode macroblock.

Solutions to the above problem existed in some early versions of the H.264 standard. The current H.264 solution is that when context-based adaptive binary arithmetic coding (CABAC) is terminated, the next bit to be decoded must be a single specific syntax element (i.e., RBSP_STOP_ONE_BIT). For example, if an

offset value (i.e., CODIOFFSET) is larger than or equal to a range value (i.e., CODIRANGE), a value of 1 is assigned to a value (i.e., BINVAL), no renormalization is carried out and the CABAC encoding is terminated. In such a case, the last bit inserted in register RBSP_STOP_ONE_BIT is the offset value CODIOFFSET.

The disadvantage of such an approach is a lack of an ability to terminate the CABAC encoding prior to sending an I_PCM mode macroblock unless the slice being encoded is also terminated at the same time. For an I_PCM mode macroblock, the next bit decoded after terminating the CABAC encoding could be either a syntax element (i.e., PCM_ALIGNMENT_ZERO_BIT), or a first bit of the syntax element (i.e., PCM_BYTE).

The existing solutions disallow CABAC encoded slices that contain any I_PCM mode macroblocks. To use the I_PCM mode with the existing solutions, the current slice must first be terminated, and then a new slice begun with an I_PCM mode macroblock.

There are typically many bits of overhead associated with terminating an existing slice and beginning a new slice since a new slice header must be transmitted. For broadcast applications in which the overhead bits needed by many small slices are not needed

for error resilience (i.e., with internet streaming applications) the existing approaches carry an undesirable penalty in overhead.

It would be desirable to implement a method and/or circuit that effectively bypasses the entropy encoding stage in an H.264 compliant CODEC for bits that belonged to an I_PCM mode macroblock.

Summary of the Invention

The present invention concerns a method for decoding a bitstream. The method generally comprises the steps of (A) generating a first signal and a second signal by parsing a common slice in the bitstream, (B) generating a third signal by entropy decoding the first signal, and (C) generating a video signal by combining the second signal and the third signal.

The objects, features and advantages of the present invention include providing a method and circuit that may (i) bypass the entropy decoding stage as needed, (ii) be used in an I_PCM macroblock, (iii) be compliant with an H.264 CODEC, and/or (iv) efficiently decompress bitstreams containing CABAC mode and I_PCM mode macroblocks.

Brief Description of the Drawings

These and other objects, features and advantages of the present invention will be apparent from the following detailed description and the appended claims and drawings in which:

FIG. 1 is a block diagram illustrating various components of a compressed video transmission system;

FIG. 2 is a block diagram of an encoding system;

FIG. 3 is a block diagram of an decoding system;

FIG. 4 is a block diagram of an arithmetic coding engine; and

FIG. 5 is a flowchart of a decoding decision logic.

Detailed Description of the Preferred Embodiments

Referring to FIG. 1, a block diagram of a system 10 is shown. In general, a content provider 12 presents video image, audio or other data 14 to be compressed and transmitted to an input of an encoder apparatus 100. The compressed data 18 from the encoder 100 may be presented to an encoder transport system 20. An output of the encoder transport system 20 generally presents a signal 22 to a transmitter 24. The transmitter 24 transmits the compressed data via a transmission medium 26.

On a receiving side of the system 10, a receiver 28 generally receives the compressed data bitstream from the transmission medium 26. The receiver 28 presents a bitstream 30 to a decoder transport system 32. The decoder transport system 32 generally presents the bitstream 30 via a link 34 to a decoder apparatus 102. The decoder 102 generally decompresses the data bitstream 30 and presents the data via a link 38 to an end user 40.

The present invention may be implemented in the encoder 100 and/or the decoder 102. The encoder 100 and/or the decoder 102 may provide up to three different syntax elements that may follow a CABAC termination. The decoder 102 may marginally increase processing complexity since the syntax element that follows a CABAC termination is no longer always known without additional contextual information.

A section of the current H.264 specification applicable to the present invention states: "If codIOffset is larger than or equal to codIRange, a value of 1 is assigned to binVal, no renormalization is carried out and CABAC decoding is terminated. When decoding end_of_slice_flag, the last bit inserted in register codIOffset is rbsp_stop_one_bit.

The present invention implements a decoder 102 capable of parsing one of three potential sytnax elements (e.g., RBSP_STOP_ONE_BIT, PCM_ALIGNMENT_ZERO_BIT, or PCM_BYTE) following a CABAC termination. The present invention enables a valid syntax that may be produced by the encoder 100 to produce a bitstream with slices containing I_PCM mode macroblocks in slices that switch from non-I_PCM mode macroblocks to I_PCM mode macroblocks (in macroblock scan order). The present invention may be used to produce efficiently compressed bitstreams containing I_PCM mode macroblocks.

While the present invention is described in the context of H.264/MPEG4-AVC encoders, decoders, and transcoders, other implementations may be implemented. For example, the present invention may potentially be used in future video standards that incorporate (i) an arithmetic entropy coder and (ii) an PCM macroblock mode.

Referring to FIG. 2, a diagram of the encoder apparatus 100 is shown. The encoder 100 generally comprises a block (or circuit) 104, a block (or circuit) 106, a block (or circuit) 108, a block (or circuit) 110, a block (or circuit) 112, a block (or circuit) 114, a block (or circuit) 116, a block (or circuit) 118,

and a block (or circuit) 120. The block 104 generally parses a video, audio and/or image data signal (e.g., IN). The block 106 generally provides subtraction of temporal and/or spatial and/or inter-band prediction(s) to remove redundancy. The block 108 generally performs a transform and quantization process (if needed). The block 110 generally performs an inverse quantization and delay process (if needed). The block 112 generally performs a zig-zag scan (or other serialization) of two dimensional data and binarization for binary arithmetic encoding. The block 112 only performs the serialization functions if needed on a particular bitstream. The block 114 generally performs arithmetic entropy encoding (AC). The block 116 generally performs pulse code modulation (PCM) encoding. If the output data from the block 104 is already PCM data, the block 116 simply passes the PCM data to the block 118. The block 118 chooses either arithmetic entropy coded data or raw PCM data. The block 120 presents a compressed bitstream (e.g., COMP).

Referring to FIG. 3, a diagram of the decoder apparatus 102 is shown. For H.264/MPEG4-AVC the choice between arithmetic entropy coded (AC) and PCM data may be made on each macroblock of data. In general, the decoder 102 reverses the steps performed by

the encoder 100. In particular, the decoder 102 generally comprises a block (or circuit) 130, a block (or circuit) 132, a block (or circuit) 134, a block (or circuit) 136, a block (or circuit) 138, a block (or circuit) 140, a block (or circuit) 142, a block (or circuit) 144 and a block (or circuit) 146. The block 130 generally receives the compressed bitstream COMP (e.g., from the block 120). The block 132 generally parses the bitstream COMP by choosing arithmetic entropy decoding or presents raw PCM data of each block of data as signaled in the compressed bitstream COMP. The block 134 generally performs arithmetic entropy decoding (AC). The block 136 generally performs pulse code demodulation. If the data signal is to remain as PCM data, the block 136 may simply pass the data to the block 146. The block 138 generally performs inverse binarization for binary arithmetic decoding and inverse zig-zag scan or other two-dimensionalization of the serial data. The block 138 may be an optional block. The block 140 generally performs inverse transform and quantization (if needed). The block 142 generally provides the delay of the output of the block 140 to present a separate input to the block 144. The block 144 generally performs additional temporal and/or spatial and/or inter-band prediction(s) to restore redundancy. The block 146 generally

combines the signals generated by the block 136 and the block 144 to present a video/audio and/or image data signal (e.g., OUT).

If the switch between entropy coding AC and PCM data were on the slice or picture level, there would be no need for the current invention. For example, in the H.264/MPEG4-AVC standard, the AC data would always be properly terminated through the pre-existing process for end-of-slice termination in the standard. Such termination would allow a clean and error-free transition to PCM data for the blocks of data following the termination.

This invention is generally implemented in the Arithmetic Entropy Coding/Decoding (AC) blocks of both the encoder 100 and/or the decoder 102. The present invention provides a new method for determining when to renormalize the arithmetic coding (before termination) when encoding/decoding the END-OF-SLICE-FLAG and the BIN-INDICATING-I_PCM mode flag.

In the new method, the conditions for performing renormalization and setting the current value BINVAL in the arithmetic encoder/decoder 114 and/or decoder 134 are the same for termination following either the END-OF-SLICE-FLAG or the BIN-INDICATING-I_PCM mode. By comparison, in the old method, the value BINVAL could be set to 1 and renormalization not performed only if

the last bit inserted in the register CODIOFFSET was RBSP_STOP_ONE_BIT (i.e., only for encoding/decoding the END-OF-SLICE flag). For the old method, the value BINVAL could never be set to 0 and renormalization performed for encoding/decoding the BIN-INDICATING-I_PCM mode. The old method lacked a correct termination of the AC engine for encoding/decoding the BIN-INDICATING-I_PCM mode, preventing the possibility of macroblock-level switching in the middle of a slice (a group of macroblocks composing a portion of one video frame or field) of data from arithmetic coded macroblocks to PCM coded macroblocks.

Referring to FIG. 4, a simplified block diagram of a binary arithmetic coding engine 150 is shown. The engine 150 generally comprises a block 152, a block 154, a block 156, a block 158, a block 160 and a block 162. The block 152 generally receives the value BINVAL (which is the current binary input symbol from the binarization block of FIG. 2). The block 152 may also generate a context adaptive binary arithmetic encoder (CABAC) derived input (e.g., CTXIDX). The derived input CTXIDX may be a current context, which is derived from the history of the bitstream (e.g., from other symbols that have passed through the AC encoder previously).

The block 154 may generate internal state variables in registers of the AC encoder (e.g., CODIOFFSET and CODIRANGE). The block 154 may also execute internal procedures to modify the state of the registers and to renormalize the contents of the registers when necessary.

The block 156 may write bits to the compressed bitstream of FIG. 1, taken from the register CODIOFFSET. The block 158 may receive bits read from the compressed bistream of FIG. 3, and added to the register CODIOFFSET. The block 158 may also generate the CTXIDX internally derived input, which is derived from the history of the bitstream (e.g., from other symbols that have passed through the AC decoder previously).

The block 160 may generate the internal state variables in the registers of AC the decoder (e.g., CODIOFFSET and CODIRANGE. The block 160 may also execute a procedure to modify the state of the registers and to renormalize the contents of the registers when necessary. The block 162 may generate the value BINVAL as the current binary output symbol to the binarization block of FIG. 3.

The present invention is not necessarily limited to use only for binary arithmetic encoding, but may also apply to non-binary arithmetic encoding. However, non-binary encoding may

result in the BINVAL being replaced with another value (e.g., SYMBOLVAL). Various other changes may also be implemented to the internal working of a non-binary arithmetic coding engine.

Referring to FIG. 5, a flowchart of a method (or process) 200 is shown. The method 200 generally comprises decoding a decision before CABAC termination that may be used in the arithmetic coding/decoding engines of FIGS. 2 and 3. The method 200 generally comprises a state 202, a state 204, a decision state 206, a state 208, a state 210, a state 212, and a state 214. The state 202 generally performs a decode terminate function. The state 204 generally decrements the first code range (e.g., CODIRANGE) by a value of two. The decision state 206 determines whether an offset (e.g., CODIOFFSET) is greater than or equal to the code range CODIRANGE. If so, the method moves to the state 208 where the value BINVAL is set to 1. The method then moves to the done state 214. If the decision state 206 determines that the code offset CODIOFFSET is not greater than or equal to the code range CODIRANGE, the method moves to the state 210. The state 210 sets the value BINVAL equal to 1. The state 212 provides a renormalization. The method then moves to the done state 214.

The method 200 may be invoked when encoding/decoding the END-OF-SLICE-FLAG or the BIN-INDICATING-I_PCM mode. The method 200 generally illustrates arithmetic decoding, which is the standardized portion of the codec for H.264/MPEG4-AVC. In the state 206, an internal state register CODIRANGE is first decremented by 2, then compared with the register CODIOFFSET. Depending on the result of the comparison the binary symbol value BINVAL is output with either a value 0 (after which renormalization of the internal state of the arithmetic coding engine must occur) or a value 1. Note that it is in the method of operation of the decode terminate process that the current invention differs from conventional approaches: namely prior art permitted only the 'NO' branch of decision block 206 to be taken for decoding of the BIN-INDICATING-I_PCM mode. In contrast, the conventional approach states that if the office value CODIOFFSET is larger than or equal to the range value CODIRANGE, a value of 1 is assigned to BINVAL, no renormalization is carried out and CABAC decoding is terminated. In conventional approaches, the last bit inserted in the register CODIOFFSET is RBSP_STOP_ONE_BIT. In the new invention other values (bit patterns) may be inserted as the last bit in the register CODIOFFSET, without impacting conformance to the H.264 standard.

The effect of the additional bit patterns is that correct termination (e.g., decoding of the BINVAL symbol and renormalization of the internal state registers of the AC engine) may now be accomplished when decoding the BIN-INDICATING-I_PCM mode.

The RenormD block 212 is a process that generally changes the values of CODIOFFSET and CODIRANGE dependent only upon the current values, and additional bits that may be added to the register CODIOFFSET (read from the bistream) during the operation of the process.

While the invention has been particularly shown and described with reference to the preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made without departing from the spirit and scope of the invention.